COVER SHEET

10th International Command and Control Research and Technology Symposium

Model-based Communication Networks and *VIRT*:

Filtering Information by Value to Improve Collaborative Decision-Making[1]

Rick Hayes-Roth
hayes-roth@nps.edu
Professor, Information Sciences Department
Naval Postgraduate School
Monterey, California
April, 2005

Abstract

Command-control and other distributed, collaborative systems should achieve the best possible results with resources available. We should measure these systems in terms of the *quality* of decisions made. *Better* decisions lead to *better* outcomes, because superior choices are made about what to do, with what assets, where and when. Just as we measure manufacturing processes in terms of *value added* at each stage, we want each processing step in distributed collaborative operations to maximize the ratio of added value to cost. Both computerized agents and human personnel receive information from others, process it, and then produce additional information for others downstream in the operational processes. This paper shows that current architectures do not promote high productivity. Specifically, most current approaches encourage an increase in information supply and exchange *per se*, producing *glut* rather than *value*. This paper explains how we can significantly increase the productivity of each operator and the success of overall missions. The approach, called VIRT, treats collaborators as participants with shared models. These models determine which information is high value and for whom. The architecture gives priority to conveying high value information. Similarly, low value bits are filtered out, saving resources and optimizing value attained.

[1] This work was supported by a research initiation grant from NPS to the author.

Introduction and Overview

Every modern organization aspires to improve its performance through better use of information technology. Organizations seek to increase their agility and make better, more adaptive responses to changing circumstances. As communication technology improves, organizations can operate over wider distances and can even assemble operational components on an ad hoc basis to meet requirements of a specific objective. People and machine-based agents operating in these organizations must collaborate with other members of their mission-oriented teams. They send information to and receive information from each of their collaborators, and each collaborative team aims to process this information to reach an effective and timely decision. After Boyd[1], the cycle of information processing and decision-making is often referred to as an OODA-loop. To be effective competitors, organizations must close these loops faster than their opponents, so that they can drive rather than react to the opponents' behaviors.

Several trends work against our ability to close our decision loops quickly. First, the number of networked collaborators is increasing, meaning that we must process information from and for an increasing number of partners. Second, the number of relevant information sources and quantity of availability data are both increasing rapidly. Third, the times available for decisions are shrinking as we seek to compete with more agile adversaries. Where cycle times in the military were traditionally anywhere from 24 to 72 hours, our aspirations are now to identify, analyze, decide, and prosecute some targets within a few minutes. Thus, making information more available and increasing its flow among collaborators ultimately reduces the quality of decisions. Just as time-sharing computers "thrash" when they become overloaded with pending tasks, people can't make good decisions when they are time-stressed and overloaded with information waiting for their attention.

The purpose of this paper is to describe a fundamental shift in the way we approach communication among time-stressed collaborators often glutted with information. The new approach is called a "model-based communication network" (or MCN). Collaborators who employ an MCN can drastically reduce the amount of work required and can significantly increase their information-processing productivity. The major benefits of MCNs are delivered through services that filter what information people and machine agents receive. We call the services that deliver *valued information at the right time* VIRT, for short. VIRT services essentially filter information so that high-valued bits are prioritized and low-valued bits are deprecated or withheld. In this way, each collaborator's incoming queue of messages is dynamically prioritized, enabling the person or agent to work on the most important information first. This "best first" approach produces the productivity gains organizations need to thrive in a networked, information-rich environment.

This paper introduces a component-based architecture for VIRT and illustrates it with examples. I describe nine VIRT components and their interconnections. After that, the paper discusses related research, current challenges and opportunities, and conclusions. This paper should prove helpful to architects, designers and implementers of systems to support network-centric operations or any environment for time-stressed collaboration and decision-making.

The Basic Problem with Current Approaches: Stateless Networking

The modern military, as other modern extended enterprises, aims (1) to exploit superb information (2) to achieve unprecedented levels of effectiveness through (3) agile, coordinated control of resources. These new enterprises form virtual organizations on an *ad hoc* basis, quickly assembling resources with needed capabilities and integrating them into a unified operational federation. To succeed, these organizations must collaboratively construct and consistently maintain a shared understanding of several important things: mission intent; alternative plans under consideration and those being executed; and the evolving situation, which includes the past, current, and future expected position and status of all relevant entities in the environment. The term common operational picture (COP) is sometimes used to mean this shared model of the battle space, but it usually connotes a more limited *view*. The term *world model,* meaning all of the facts and beliefs about the environment, is

more apt. The key capability required to enable virtual organizations to coordinate and execute at maximum effectiveness in dynamic environments is a *shared world model.* Any attempt to lay a new foundation for collaborative networks should be driven by this requirement, and putative improvements should demonstrate how they raise the quality of the shared understanding that enables synchronized, coordinated, intelligent real-time decision-making and control.

Conventional approaches to communications have focused on laying pipes that move bits using *stateless* protocols. *State* refers to what a system remembers about what it has already done and that causes it to behave differently going forward. Stateless communication is very appropriate when we are focusing on connecting mostly independent entities, for limited interactions, which arise pretty much randomly. Whatever memory is required in these interactions is supplied by the interacting entities. Usually that means two communicators begin by establishing their identities and their credentials, and then they begin to work on establishing shared state. This requires that they describe their current beliefs, identify discrepancies, and determine how to resolve those. From that point on, as long as they stay synchronized, they can quickly process new reports by a kind of triage: repetitive and redundant information can be discarded; confirming information can be coalesced into the models that "explain" it; new but unsurprising information can be accepted and used to augment to the current model; and disconfirming information can be subjected to further tasking and analysis, as appropriate. This is the maximum possible level of efficiency for handling information communicated between two parties.

Unfortunately, the actual situation in most military operations is much more complicated and much less efficient than in this idealized two-party on-going communication. Rather than maintaining continuous shared state, the communication is usually stateless. The parties don't stay in continuous synchronized dialog. They effectively "hang up" after each short transmission. They send messages whenever they think they have information worth reporting. The longer the parties operate independently, the more their respective world models diverge. Each time a recipient receives a message, the recipient must now also attempt to determine whether differences in the senders' world models affected what they've said, why they're saying it, and how best to interpret and utilize their statements. Moreover, communications in net-centric organizations are not merely 1-to-1, but n-to-n, meaning that each party is receiving messages from others whose own states relativistically affect what they're transmitting. Absent an absolute, common frame of reference, each communication requires the recipient to try to determine the meaning, relevance, validity, and significance for its own world model. In the fog of war, this process inevitably results in these problems: (1) many messages are sent repeatedly; (2) many recipients are overloaded; and (3) many incompatible and inconsistent views are held by different parties. The process is grossly inefficient.

The ideal communication framework for network-centric organizations is like the *blackboard* architecture[2], in which each actor can see all previous inferences and all important ideas are woven into a structure of shared beliefs. In the original blackboard architecture, the posted ideas were called *hypotheses* and these were *linked* to represent various kinds of supporting logical relationships. Publish-subscribe architectures[3] are simplified abstractions of the blackboard. In these, recipients identify what information they are interested in, and the system routes matching items from publishers to them. *Distributed blackboard* architectures are actually the best model of the ideal communication framework[4, 5]. In these, copies of subsets of the logically global blackboard are distributed to provide fast local caches for each participant, and various processes are employed to keep the replicas synchronized and consistent.

In extended and net-centric enterprises, collaborators need to share beliefs that consistently reflect their individual roles in collective plans and operations. *Plans* are an example of shared decision products best modeled as *compound objects*. These contain constituent objects that describe the elements of a plan, such as each aircraft's mission, route, targets, refueling, weapons, etc. In addition to plans, the organizations need to share compound objects that describe their *situation analyses*, including status and capabilities of blue and red forces, terrain, networks, and so forth. Each

participant in intelligence, plans, and operations should be able to see a permitted view of current beliefs and should be able to make incremental adjustments to those when they have new information. Changes in beliefs should be propagated to replicas of the corresponding objects. When changes in some beliefs undercut current plans, either by nullifying some prerequisite or altering the relative desirability of a previously rejected alternative, this condition should trigger a process that reassesses the affected plans and associated analyses. By maintaining state, important news can be automatically detected in many cases, and this can allow the responsible parties to focus attention exactly where it's needed.

The appropriate model for net-centric collaborative organizations should recognize their essential nature: they must be continuously synchronized though distributed, and they must be driven by significant events, those corresponding to changes that have material impacts on on-going plans. Collaborators meld and share belief structures that describe the environment, resources, capabilities, plans, and expected results of plans. These belief structures correspond to compound objects with support relationships. The communications that people should experience, because these are the ones that matter, are those that signify a material change in beliefs. Other communications should be largely invisible, as they work mostly autonomously to maintain distributed, synchronized state. Thus effective collaborative problem-solving requires: (1) the ongoing, mostly unconscious, maintenance of melded world models; and (2) the event-driven, conscious assessment of how real "news" affects previous decisions and choices for future actions. In short, models enable us to know which bits convey information because they are "news," and we must give priority to shipping those bits to consumers who value them. Knowing how "news" changes expected outcomes for various potential consumers enables us to target the news to the right consumers promptly.

VIRT Improves Time-stressed Collaborative Decision-making

DoD has committed to transform around concepts of Information Superiority (IS) and Network Centric Operations and Warfare (NCOW)[6]. FORCEnet, as an example, aims to provide the Navy the capabilities required to support agile, rapid, precise, effective and efficient planning and operations. In these new concepts, warfighters can access and employ whatever information they need to perform their tasks. In short, every person should operate on the right information. One problem, however, is information glut. Too much information is available today, and the problem grows worse over time. Another problem is that people have to work hard to find the valuable information, either because it doesn't automatically find them or because it's buried amidst megabits of data and messages that are not important for their particular mission concerns.

Thus, IS/NCOW depends on enabling each individual to receive valuable information at the right time and, in parallel, the automatic filtering out of low-value information. This requires improved means for allowing the needs of individuals to determine just what information finds them, so they can spend more of their time assessing and acting upon high-value information. This would have the effect of increasing individual productivity throughout the military and, as a consequence, help attain the goals of IS and NCOW. Without such a capability, moreover, increasing information loads will have the paradoxical effect of reducing mission effectiveness.

To solve these problems we need a model-based communication network (MCN) that delivers to each of its customers tailored products that satisfy the objectives of "valued-information at the right-time" (VIRT). The basic VIRT method adapts the information flow around an understanding of mission plans, their rationales or *justifications*, the assumptions and forecasts they depend upon, and their expected outcomes. In short, VIRT looks for information that materially affects expected outcomes and communicates that to decision-makers so they can consider and adopt preferable alternatives in a timely way.

Here's a simple example from aviation, where a pilot's route is planned at low altitude over low mountains. The planner considers many variables, constraints, and outcome possibilities in selecting an optimum route. For one type of mission, the goal may be the shortest flight; for another it might be

the stealthiest flight; and for another it might be one with best line-of-sight communications to several parties along the route. The types of variables that must be considered include: terrain elevation; winds aloft; fuel consumption and capacity; routing waypoints and their relationship to other parties in the environment; precipitation and temperature; etc. Constraints include, for example: the flight must not consume all the fuel and, in fact, the planned flight must allow for an additional hour of emergency reserve; the flight cannot encounter icing and allow for ice accumulation; the flight must maintain safe clearance above terrain, especially when winds and steep terrain interact; etc. The planner considers many alternatives for the future flight, using forecast weather data and other information and assumptions. In light of the mission objectives and assumed information, the planner chooses the best alternative for the flight plan.

Continuing with our example, considerable time usually passes between the moment when a flight is planned and when the flight actually begins. In some cases hours or even a day or more might pass. As time passes, the information available evolves and changes. Forecasts improve as their distance in the future shrinks; in addition, direct observations supply facts for what previously required assumptions. Information continues to flow into organizations like the Navy's Fleet Numerical Meteorological and Oceanographic Command (FNMOC[2]) right up to the aircraft's departure and throughout the planned flight. FNMOC "knows" when weather data, for example, differ from what had been previously forecast or assumed. Enabling FNMOC to "know" which changes are significant to the pilot will then enable it to supply valued information at the right time, i.e. implement VIRT services.

In short, the supplier of information bits needs to understand its customer's sensitivities. In this example, a change from a low probability of enroute icing to a substantial probability of enroute icing would be material to the pilot. Assuming the currently preferred plan didn't violate a "no flight allowed into icing" constraint, a newsworthy violation of that constraint could arise if the forecast changes to anticipate a combination of sub-freezing temperature and visible moisture along the planned route at the planned flight time. For FNMOC to implement VIRT, it needs to know the planned route and time, constraints on acceptable flights, and a way to convey news to the operator. Each operator and plan can have its own sensitivities. Aircraft at high altitude usually are above the weather and have de-icing equipment, so they are unlikely to consider precipitation and subfreezing temperature important. On the other hand, their flight levels are more affected by "jet stream" winds and these can significantly affect fuel consumption, as just one example.

So the essence of VIRT is knowing which consumers really care about what news. Suppliers of information should monitor for a change in their information (news) that would interest operators, because it changes their beliefs about expected outcomes. The final element of VIRT consists of the conveyance employed to transmit the valued news to the user. This should include a means to highlight news in an appropriate way. Preferably, the highlighting causes recipients to attend to news with a priority that closely approximates the importance they attribute to it. Urgent and vital information deserves high priority. Unimportant data and stale information deserve low priority.

We can employ a range of possible methods to implement the essence of VIRT. In the ideal world, the plans and plan evaluation methods of the operators might be known to the information suppliers. Then whenever a supplier noticed a change in relevant information, it could "simulate" the operators' thinking to determine whether the operators would alter their previously selected plans. In just those cases, it would alert the operators. Otherwise, it would not bother to pass along insignificant changes. As a much more modest objective, we have chosen to allow operators to tell us what kinds of changes

---

[2] I collaborated with FNMOC on the implementation of VIRT for their customers. I have been fortunate to have the knowledgeable and enthusiastic support of the former FNMOC Commander, Chris Gunderson (CAPT USN RET), and several of his talented staff, including: Bruce Gritton, FNMOC-CIO; Ensign Darin Keeter; and Doug Gentges, a contract architect/programmer. Gunderson is now Executive Director of the World Wide Consortium for the Grid (W2COG), where VIRT is a major architectural tenet. See www.w2cog.org .

are significant to them in light of specific plans they have considered and selected. The VIRT service then takes responsibility for monitoring the identified types of changes and conveying them promptly to the interested parties.

Even this modest ambition for an initial VIRT service will produce substantial benefits. Every planner and operator is sensitive to some types of potential changes throughout the period leading up to and during plan execution. A typical pilot for a simple mission, for example, may be responsible for monitoring dozens of information types throughout a 12-hour period. An entire organization or a coordinated military mission force, for example, can make millions of "reads" to keep their plan justifications fresh. Usually, nearly all of those justification-maintaining examinations will turn up nothing valuable. As a consequence, the rare and important deviation from acceptable condition will often be overlooked.

To make VIRT and model-based communication networks routinely available, we need to provide some generic capabilities that enable suppliers and consumers of information to understand what information is valuable. These generic capabilities can then be specialized for particular domains of application and communities of practice, as when weather specialists and aviators establish a shared understanding of concerns such as "enroute icing" and "headwinds." In the next two sections, I explain what these generic capabilities are and illustrate how we specialize them for particular applications.

Overall Technical Strategy and High-level Architecture

In this section, we consider a high-level architecture for VIRT that exploits a set of semantic models that describe the information suppliers make available to consumers. The simplified architecture is illustrated in Figure 1.

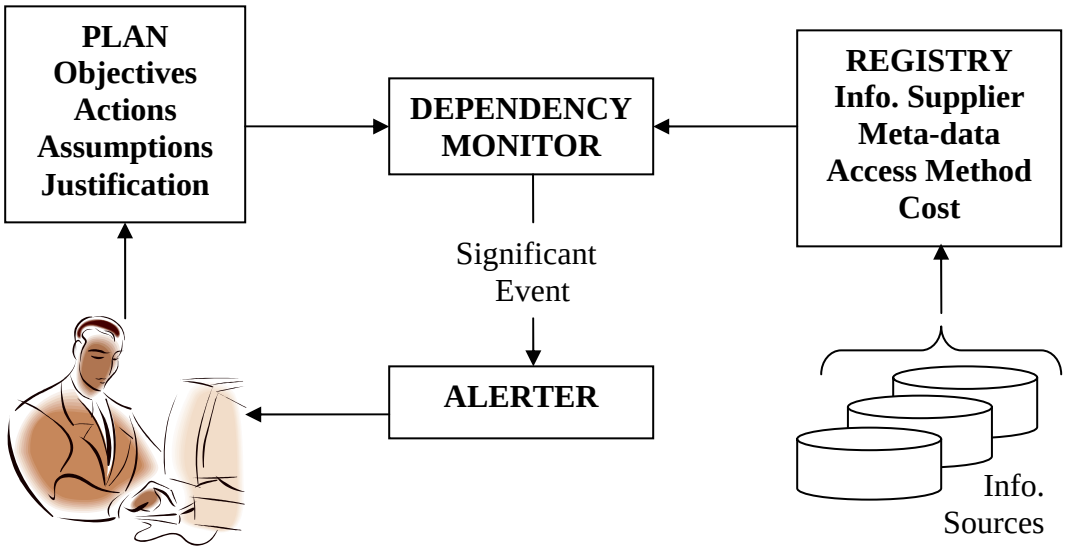

Figure 1. A simplified architecture for VIRT.

The architecture in Figure 1 simplifies much of the complexity by focusing on just a single plan, apparently planned and periodically adjusted by only the one person illustrated. Of course, real organizations comprise many teams pursuing many objectives, so there are many planners and plans extant at any point in time. Nevertheless, the key elements of the VIRT approach appear in this simple view.

The overall flow in Figure 1 starts on the left side, where a plan has been generated. Each plan describes time-phased actions that should accomplish the plan's objectives. The planner considered what the state of the world would be at the time the plan executes, and his/her beliefs about that future state correspond to the "assumptions" recorded in the plan. When planners select a plan, they usually

evaluate it and compare its costs and benefits to other alternatives. They can record their reasons for selecting one particular alternative in the form of a *justification*. A justification ordinarily explains how the planned actions should accomplish the objectives in a situation where the assumptions validly hold and, also, why the planners prefer the selected plan to the alternatives they considered. The justification often reflects that all of the alternatives considered had excessive risks or costs in comparison to the chosen one.

Let's consider a simple example. The planners might have an objective to rescue a small group of people on the ground in forested terrain. Their basic choices consist of reaching the people by ground or air and extricating them by ground or air. The likely options for ground transport include wheeled and tracked vehicles, horses, and humans. The likely options for air transport include rotorcraft v. fixed wing aircraft. Given a number of factors, they quickly reject all but the following skeletal plan:

1. Survey the area by fixed wing aircraft to find the best landing spot for a helicopter.
2. Send a helicopter with a search and rescue (SAR) team.
3. Land the helicopter at the chosen site.
4. Find and recover the party using the SAR team.
5. Depart by helicopter and return the party to a chosen medical facility.

Given this skeletal plan, the planners then focus on possible aircraft and routes, total expected flight times and associated fuel requirements, and possible time sequences for the flights. The flights become highly dependent upon the assumed wind, visibility, and icing conditions en route and at the search and rescue (SAR) area.

Let's complete the example plan. The planners assume that a 90-minute aerial survey will be required to choose the best landing site. They choose an available low-altitude aircraft that carries appropriate instruments and can reach the site with only a two-hour flight. The aircraft has adequate fuel for 6.5 hours which is just sufficient for two 2-hour legs, a 1.5 hour survey, and still leaves a 1-hour mandatory reserve. The winds in the area are forecast to be excessive between the hours of 1300 and 1800 local, and adequate sunlight is expected only from 1000 to 1900. For these reasons the flight is planned for early tomorrow morning, so that the survey begins promptly at 1000. Thus, take-off is scheduled for 0800. The helicopter is scheduled for a 3-hour flight to the search area, and is planned to depart at 0900, so that it can receive landing coordinates at 1130 from the SAR aircraft survey team 30 minutes prior to touching down.

Even this example leaves out countless details, but it provides enough to illustrate the key VIRT architecture features. The VIRT dependency monitor takes the responsibility to watch for changes in forecast or actual conditions that threaten the plan by undercutting its justification. In the current case, the monitor needs to revalidate periodically the key assumptions regarding aircraft availability, aircraft capabilities, winds, visibility, icing and fuel consumption. Table 1 shows a sample of possible changes in the world or our model of it that might undercut the plan's justification and, for each, one or more information sources that the monitor needs to reassess periodically so it can re-assure the justification.

This table lists seven out of hundreds of possible vulnerabilities of the example plan. Aircraft often have maintenance problems that ground them. If either of the planned craft are grounded before the operation is complete, the whole plan may fail. Therefore, planners must continually monitor the readiness of the craft. Similarly, the survey mission assumes the availability of various instruments, and these must continue to function until an adequate helicopter landing area is selected. Thus, mission planners should monitor and revalidate the equipment's functionality. The third item supposes that an aircraft substitution has occurred, as in response to a problem like the first or second ones discussed. In this case, the new aircraft must have as much range, load, and instrumentation capabilities as required of the one it replaced.

The fourth and fifth problems challenge the ability of the aircraft to complete the planned flights, either because the new conditions might require excessive fuel or prohibit flight. These possibilities always exist, though at varying degrees of likelihood depending on locale and season. Nevertheless,

the plan is vulnerable to changes in these meteorological conditions, so the dependency monitor should continually revalidate the ability of the aircraft to fly the planned route, maintain an adequate fuel reserve, and avoid flight into icing conditions.

Table 1. Vulnerabilities and the associated dependencies monitored.

Changes that Undercut Plan Justification	Dependency that Must be Monitored
1. Planned aircraft down for maintenance	Readiness of planned aircraft
2. Survey instruments inoperative	Readiness of planned survey equipment
3. Substitute aircraft has reduced capability	Capacity and capability of replacements
4. Increased headwinds eliminate fuel reserve	Winds aloft and fuel consumption
5. Enroute icing reported by other aircraft	No icing conditions observed or forecast (temperature, precipitation, ice)
6. Survey team finds no suitable landing area	Survey objective accomplished satisfactorily
7. Departure of survey aircraft delayed	Survey objective accomplished on time

The sixth and seventh problems undercut the logic of the plan, by making it impossible for the helicopter to depart at a fixed time, receive coordinates of a landing site en route, and land shortly thereafter to perform the rescue function. The system should monitor the continuing plausibility of the assumptions the helicopter plan depends on, including the timely and satisfactory completion of the survey objective to find a landing site in time.

While there are many other ways this plan, and any plan, can be thwarted by violations of explicit or implicit assumptions[7], the point here is that machines should be employed to monitor as many important dependencies as possible. When unfolding events violate any of these key assumptions, planners should consider the consequences. The VIRT system monitors such dependencies and alerts planners when significant events occur. These significant events include the types listed in Table 2.

Table 2. Significant Events Types and Illustrative Examples.

Category of Significant Event Monitored	Illustration from Example Plan
Unavailability of required resource	Aircraft not grounded for maintenance
Inadequate capability of employed resource	Aircraft instruments operative, load adequate
Adverse change in forecast weather	Increased headwinds forecast en route
Adverse observations reported	Other pilots report high winds aloft and icing
Plan's justification negated or nullified	Resources, capabilities, and time intervals now available probably can't achieve an objective

VIRT works by seeking significant events in dynamic data sources. To do this, it must understand what significant events undercut assumptions the plan depends on and how to access and query information sources for the events of interest. The illustrative examples in Table 2, for example, might be monitored by periodically examining aircraft readiness and capability databases, wind and icing pilot reports, wind and icing updated weather forecasts for the airways and times of our intended flights, and expected start and completion times for various planned tasks that other tasks depend upon. Given a very specific list of significant events, plan dependencies, and information sources, a specialized VIRT application could be easily designed and straightforwardly implemented. We would still need to specify the best way to alert the human planners when we detected particular categories of significant events. For example, if we computed that new wind forecasts undercut the ability to maintain an adequate fuel reserve, we'd want the specialized application to consider and compute some potential workarounds, such as changes in route, atltitudes, or time. The specialized VIRT application could then offer more than just "a new problem"; it could also constructively suggest a

potential adaptive response. An excellent example of a specialized application prototype that addresses many of the challenges of monitoring weather for significant events and determining when and how to alert pilots is the AWARE system, described by Ruokangas and Mengshoel[8].

The simplified architecture we are describing is aimed at solving a broader generic problem, so that almost any planner can employ it for any kind of plan by seeking significant events among any pertinent information sources. Rather than a specific application, therefore, the VIRT architecture aims to provide a generic service for plan monitoring and for intelligent filtering of potentially relevant, dynamic information sources. In the generic architecture, the dependency monitor can infer what types of events to look for from any plan whose components include assumptions and a justification. The monitor can also be advised by an operator how to focus or optimize its functions. VIRT also employs a registry of available information sources to exploit whatever sources become available. VIRT is open to new information suppliers, who need only describe what their information sources are, how to access and query them, and how to reimburse or compensate the supplier if payment is required. Lastly, the architecture is open on the question of how alerts of significant events should be communicated. We expect there will be a variety of alerting methods, some more appropriate than others for particular types of users, tasks, information sources, mission objectives, significant events and equipment contexts.

In addition to the type of example plan we discussed in this section, our work on VIRT with FNMOC currently focuses on two particular Navy operational scenarios. The first of these addresses the problem of assuring that submarines receive high-value information over their limited bandwidth channels. To do that, VIRT notices when dynamically changing data differ from previously conveyed information and then determines which changes have significant import for the sub given its current mission and plans.

The second Navy scenario we are working on addresses the question of helping special operations forces minimize their risk of detection and level of effort to penetrate an enemy's defenses by minimizing their exposure to radar. In situations where the detection capability of radar depends of meteorological and oceanographic conditions, our VIRT application determines when weather and sea state change enough to justify replanning, and then triggers that replanning. In this way we hope to make SEAL missions less risky, less physically demanding, and more adaptive to dynamically changing environment parameters.

In sum, the high-level architecture of VIRT aims to improve group synchronization by understanding how changing situation variables affect their plan, monitoring specifically for potentially important changes, and rapidly alerting them to significant events that undercut the logic of their plans. We envision a VIRT system that is open to suppliers and consumers of information. The suppliers can describe what information they supply and how to access it. The consumers can describe what assumptions justify their plans and how deviations from those assumed conditions signify plan vulnerability and portend problems. Our initial VIRT projects don't try to automate the functions of re-justifying a plan in light of changing circumstances or re-planning a no-longer-appropriate plan. Those goals would require a narrow focus that could be addressed by programming a computer to solve that class of problem. Instead, we defer such ambitious problem-solving capabilities to later. This enables us to focus immediately on a generic, relatively straightforward service that can be employed by many planners, consuming many types of information sources.

Product-line, Component-based Technical Architecture

When you anticipate addressing a variety of similar application requirements with mostly generic software, the best approach is to create a component-based, product-line architecture[9, 10]. A product-line architecture defines a set of reusable generic components and specifies how data and control should flow among them to solve application problems. Over time, a successful architecture encourages developers to produce interoperable components of increasing quality and capability. Our

hope is that such developments will occur to support MCN in general and VIRT services in particular. In this section, I'll describe an initial component-based architecture for these purposes.

Figure 2 illustrates the principal components of the VIRT architecture. These have some similarity to the simplified view in Figure 1, but these generic components can supply VIRT services to many different parties, related to many different plans, at the same time. Each component shown is modeled as an object with some attributes and optional methods. Generic components such as those shown can be implemented in different ways, with various specializations and enhancements. I will describe the overall component collection and interactions first at a high-level, and then do a deeper dive into each one.

We expect that VIRT services will most often be delivered in the context of organizations that plan and execute missions. A Planning Toolset component represents the types of functions and results that VIRT exploits in the planning environment. Most organizations have planning tools already, so the generic capabilities here will be obtained from existing functions augmented by some new ones. The Planning Toolset enables planners to generate candidate plans, evaluate alternatives, and justify the selections they make. Key assumptions record beliefs that a group of plans take as given. A dependency analyzer identifies particular underlying beliefs that a plan's outcome seems sensitive to. A condition generator translates these dependencies into specific conditions that should be monitored, and those conditions are monitored by corresponding Condition Monitors. Example conditions would include: "no icing en route" or "adequate fuel reserve maintained throughout plan." As time passes, elements of a plan or its assumptions may need to be updated, and a plan updater does that. This is particularly relevant as plans are executing and actual results come in to replace forecasts and expected results.

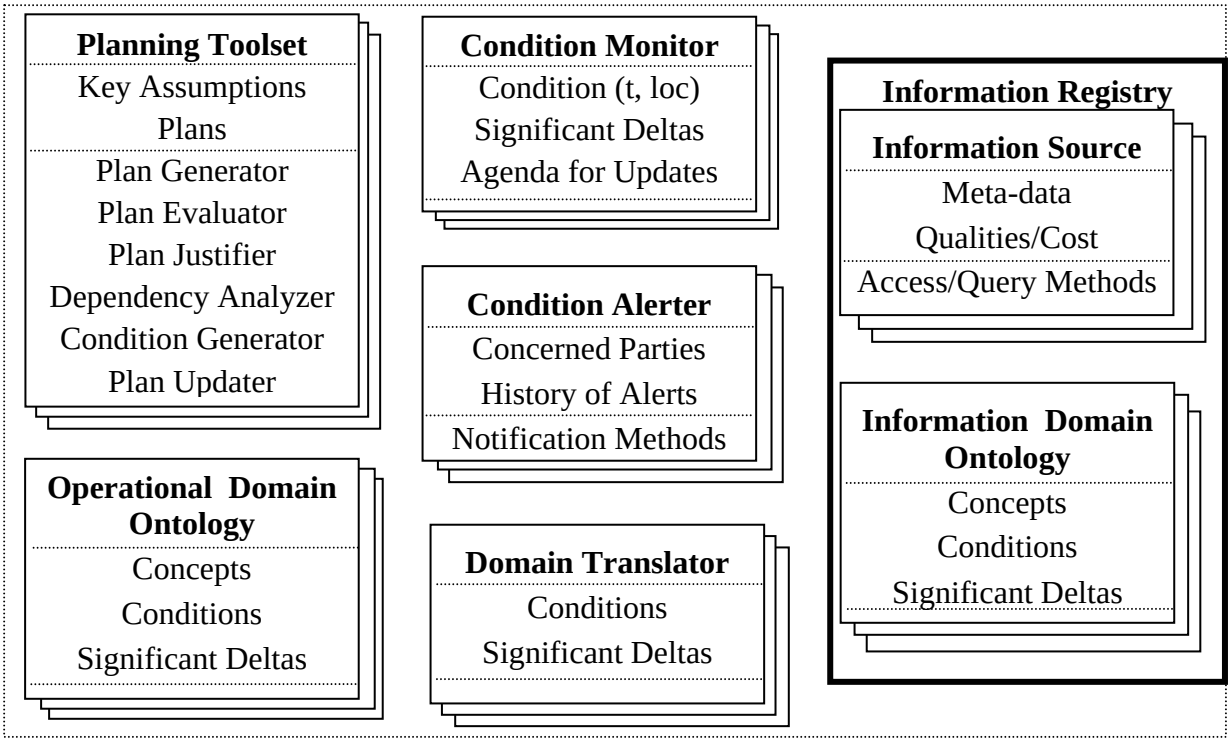

Figure 2. A component-based product-line architecture for VIRT.

Each condition generated to check and assure a plan dependency is monitored by a Condition Monitor. The condition monitor examines the value of the designated condition over appropriate time and space coordinates and records when significant changes in the value of the condition occur. It also maintains an agenda for scheduled updates to these computations. When significant events occur, as when a significant delta indicates that a condition has gone from a satisfactory to an unsatisfactory

state, the associated Condition Alerter has responsibility to notify appropriate parties who are concerned with this type of alert. The alerter can use whatever notification methods the concerned parties specified for reaching them.

The Condition Monitor obtains dynamic situation data from sources described by entries in the Information Registry. Each information source provides dynamically changing data about particular variables. The variables, encodings, and other such data definitions are described in the associated meta-data. Sources may differ in terms of their perceived or rated quality and also in terms of the cost for use. Each source provides methods for accessing its data, such as particular query languages. Usually data is characterized in terms of data dictionaries or entity-relationship models.

In the future, data will be further characterized by semantic schemas or more formal ontologies that characterize what each entity and attribute means and how different values support various kinds of inference. Ontologies will be used in two very different ways. One ontology will specify the semantics of an information source, as when an attribute such as "dew point" is explained as "a 1nm x 1nm grid of temperatures at successive 3000-foot altitude bands where airborne water molecules will precipitate into visible moisture." Another ontology will pin down the semantics the planners and operators associate with terms they've used in plans and conditions. For example, they may specify that "en route icing" means a condition of "sub-freezing temperature coupled with precipitation forecast in any area within 2 nm of the route occurring within 30 minutes of the planned flight through that area."

The last component of the VIRT architecture is a Domain Translator that can translate conditions and significant deltas expressed in one ontology into a different ontology. For example, an aviation-weather translator could translate forecast temperature and dew point information from a weather information source into "precipitation," "sub-freezing temperature," and "expected icing" that concern flight planners. In short, the Domain Translator relates concerns in the operational domain to data sources described in an Information Registry.

For each of these generic components, in turn, we'll now consider their functions, interface, interconnections, and key qualities. Any particular application could then be quickly addressed by combining and specializing available implementations of these components. Each generic component is described by a table addressing the principal facets: attributes, methods, interfaces, interconnections, and qualities. An example of each component's function is also provided. The first component considered is the Planning Toolset, in Table 3.

Table 3. Planning Toolset Generic Component Description.

FACET	VALUE	COMMENT
Name	**Planning Toolset**	Combines legacy planning aids with new methods for augmenting and annotating plans
Attribute	Key Assumptions	A list of conditions in the operational domain ontology underlying the plan
Attribute	Plans	A list of alternative plans, including actions to achieve the objective given key assumptions
Method	Plan Generator(obj)	Generates candidate plans to meet objective obj given key assumptions
Method	Plan Evaluator(p)	Assesses quality of plan p given key assumptions
Method	Plan Justifier(p)	Justifies why plan p achieves its objective given key assumptions
Method	Dependency Analyzer (p)	Determines how plan p results depend upon given key assumptions or other additional ones

Method	Condition Generator(p)	Generates conditions to monitor to assure plan p achieves expected results given key assumptions
Method	Plan Updater	Updates plans over time to reflect changes in key assumptions, actions or evaluation
Interface	User interface	Planners create plans, view expected results, designate conditions, specify alerts Planners receive alerts and view expectations and results
Interface	Machine interface	Creates condition monitor and condition alerter objects Provides access to key assumptions and plans
Interconnection	Uses operational ontology Generates condition monitors & alerters Receives alerts	Planning tools express plans in ontology terms Each plan alternative has its own conditions Each plan continually revalidated by some parties
Quality	Planning tools generate good plans VIRT services easily requested with little increased workload VIRT alerts provide concise important feedback quickly	Good plans are expected to work when executed, given assumptions already made Requesting condition monitoring and alerting should be easy with a plan and its justification and dependencies on hand VIRT reduces bit-flow to parties by filtering out frequent redundant and immaterial data
Example Use	Flight plan created with conditions monitored for start time, icing, turbulence and fuel reserves	The plan includes a route and rate of fuel consumption; route is compared to forecast weather data translated into icing, turbulence and fuel consumption values; key conditions monitored continually as time progresses

The Planning Toolset component provides a set of functions to enable planners to formulate plans, to evaluate them, to discover and select conditions to monitor, and to update the plans as things evolve over time. The toolset component provides user interfaces to support human planners and machine interfaces to create condition monitors and alerters as well as provide access to plan and assumption attributes. The toolset component should support production of good plans and enable VIRT services to monitor important conditions without a great amount of additional work on the part of the planners.

Table 4 describes the generic component for monitoring conditions.

Table 4. Condition Monitor Generic Component Description.

FACET	VALUE	COMMENT
Name	**Condition Monitor**	
Attribute	Condition (t, loc)	The condition's value at time t and location loc
Attribute	Significant Deltas	Transitions (location, time, value changes) where condition value became significant
Attribute	Agenda for Updates	Schedule to get updates from info sources
Method	Update Condition(t, loc)	Using updated data, recompute condition's value

Method	Get Update	Access an info source to get updated data
Method	Accept Update	Process asynchronous data updates received from info source
Method	Identify Significant Deltas	Determine when significant changes in the condition occur
Method	Set Agenda	Determine which info sources to access and when
Interface	Machine interface	Allow planning tools to create and modify condition monitors Allow info sources to provide asynchronous data
Interconnection	Accesses Info Registry Accesses Info Sources Use Domain Translator Signal Condition Alerter	Determines info sources to employ and how Accesses or queries relevant sources Assesses conditions and deltas in operational domain Notifies alerter when significant events occur
Quality	Effective at detecting significant events Efficient in use of costly resources	Assures vigilant assessment of conditions and detection of significant deltas Schedules info accesses intelligently, computes as needed, and only generates significant alerts
Example Use	Change in forecast headwinds implies fuel reserve will be inadequate	The operational domain condition of "adequate fuel reserve" is computed by assuring that the difference between fuel at takeoff and fuel consumed on all route legs is enough for 60 minutes of additional flight; fuel on each route leg is computed by multiplying average ground speed times fuel rate for that leg; ground speed is air speed minus headwind component

In the preceding table, we see how the generic Condition Monitor component works. One such object is associated with each condition. The condition is updated as new data are accessed or provided asynchronously. The monitor determines an efficient schedule for periodically accessing data sources. In the example, the condition "adequate fuel reserve" is monitored. The estimated fuel reserve is the number of minutes the aircraft can fly with fuel expected to be remaining after all planned route legs are flown. Standard parameters are used for fuel consumption per hour for various flight profiles. The key unknown is the amount of headwind. As the headwind forecast changes or actual headwind data become available for the planned route, the monitor recomputes the expected fuel consumption and then the amount remaining for reserve. If this becomes less than the required minimum, the condition goes from "green" to "red," and a significant delta is noted. This also means a significant event, "loss of adequate fuel reserve," must be signaled to the associated alerter.

Table 5. Condition Alerter Generic Component Description.

FACET	VALUE	COMMENT
Name	**Condition Alerter**	
Attribute	Concerned Parties	Identities/addresses of people or agents to notify
Attribute	History of Alerts	Record of notifications to parties
Method	Notification Methods	Means to convey alert to interested parties
Interface	Machine Interface	Receive significant events from Condition

		Monitor Access communication channels to convey alerts using notification methods
Interconnection	Accept significant events from condition monitor Convey alerts via communication channels	Receive, note, and disseminate criterial changes in conditions to concerned parties using notification methods
Quality	Alerts communicated promptly and concisely, avoiding annoying repetitiveness	News of significant events best conveyed to users within the context where it's most easily understood Repetition used only when acknowledgment required but not received
Example Use	The flight planner is notified with a short pop-up message that shifting winds have undercut the planned flight's fuel reserve requirement	After a flight is initially planned, the planner may be difficult to reach, because he or she may not be at the same computer where the plan was created; in addition to a pop-up message in an active planning window, instant messages, email, and voice messages might be appropriate; acknowledgment of any form should stop the alerting process

Table 6 describes the generic component for registering information sources.

Table 6. Information Registry Generic Component Description.

FACET	VALUE	COMMENT
Name	**Information Registry**	
Attribute	Information Sources	Collection of information source objects
Attribute	Information Domain Ontologies	Collection of information domain ontology objects
Method	Update Information Sources	Add, delete or modify info source objects
Method	Update Domain Ontologies	Add, delete or modify info domain ontology objects
Interface	Machine Interface	Allow access to Condition Monitor Publish updates on asynchronous channels
Interconnection	Condition Monitor reads Domain Translator reads	Monitor determines which information sources can best be used Domain translator uses info domain ontology and info source meta-data to determine which data relate to the conditions being monitored
Quality	Easy to update and administer Flexibly supports diverse and evolving sources	Registry can add, delete, and change contents without limitations Meta-data are described using flexible meta-meta-models, as are domain ontologies
Example Use	Registry incorporates three different sources on winds aloft, with different meta-data, and different	New sources of forecast and observed winds aloft are registered when available, and the condition monitor for "adequate fuel reserve" can employ whichever of these give best results for

	ontologies	acceptable cost

The Information Registry described in Table 6 above provides an open architecture for new sources of potentially relevant information. Most information sources today are described, at best, in terms of the data dictionary or database schema used to store and access data. However, meta-data including semantic schemas are increasingly available utilizing XML. Moreover, standardized language systems for ontologies, such as OWL, make it likely that more semantics and domain logic will be explicitly represented and available for use. We have anticipated this trend in this architecture component. The basic role of the registry is to provide the foundation for an "open market" of information that condition monitors can access to do their work more effectively and more efficiently. Over time, suppliers should offer new and improved sources of information that planners and operate can use to monitor key conditions more accurately and cheaply.

Table 7. Information Source Generic Component Description.

COMPONENT FACET	VALUE	COMMENT
Name	**Information Source**	
Attribute	Meta-data	Describes data types, formats, accuracy
Attribute	Qualities/Cost	Describes source's performance, reputation, etc. Specifies costs for use
Method	Access/Query Methods	Get dynamically updated data on request
Interface	Machine Interface	Allow Condition Monitor to read
Interconnection	Condition Monitor reads attributes Condition Monitor accesses & queries Asynchronous data updates to Monitor	Monitor determines which data to access, when and how Monitor uses source's methods to obtain the desired data Source uses appropriate channel to convey data updates asynchronously
Quality	Meta-data accurate Qualities/costs accurate Access/query methods reliable and produce concise results	Contents consistent with descriptions Performance consistent with descriptions Methods work as advertised and don't produce extensive amounts of extraneous bits that must be processed
Example Use	Winds aloft source	Winds aloft relevant to a route are produced by NOAA; query gives a table of wind direction and speed, along with temperature, at the time of flight, at each altitude that is a multiple of 3000', updated twice per day, reported by major air traffic regions

Table 7 describes the generic component for an Information Source. Each information source provides an independent set of data appropriate to various concerns. Typically sources correspond to periodic products of organizations such as NOAA and FNMOC for weather, and military, financial, commercial, maritime and various other products from corresponding organizations. Each Information Source describes its own data using meta-data techniques like those of XML or OMG's Model Driven Architecture (MDA). The source advertises its quality and costs, such as its reputation with consumers for timely, accurate, reliable performance. It provides ways to query and access its data. This may include techniques for posting "standing queries" that cause the source to transmit new, relevant information asynchronously to the requestor. In the example, a typical source for winds aloft is used by the monitor for the "adequate fuel reserve" condition. This source provides the wind direction and

speed in a broad area at various elevations. The fuel reserve condition would use estimated headwinds by flight phase and route segment. It might employ piecewise linear approximations based on wind forecasts from different air traffic centers the route crosses. It could also interpolate between forecast altitudes as required to match a planned flight altitude. The example might have shown another information source that could provide more precise headwind estimates or, perhaps, an accurate fuel consumption estimate based on detailed wind modeling if those were available. When new sources become available, the architecture aims to make it easy to exploit them rapidly.

Table 8. Information Domain Ontology Generic Component Description.

FACET	VALUE	COMMENT
Name	**Information Domain Ontology**	
Attribute	Concepts	Terms used and their semantic properties
Attribute	Conditions	Propositions and operators used to specify important situational characteristics
Attribute	Significant Deltas	Minimum changes in calculated conditions worthy of attention
Interface	Machine Interface	Condition Monitor may read attributes Domain Translator may read attributes
Interconnection	Condition Monitor reads attributes Domain Translator reads attributes	Monitor determines which concepts, conditions and deltas pertain to its tasks Translator maps concepts and conditions from information domain to the operational domain
Quality	Ontology expressive and interpretable Deltas as small as necessary and as large as permissible	Language used should simplify writing/editing Machines can easily perform required inferences Appropriate deltas reduce workload
Example Use	Winds aloft concepts, part of aviation ontology	Winds aloft described as a dynamic vector field over 3D space above surface; at each point, the variable has an amplitude in knots and a direction given with respect to true North; field is approximated by a grid, comprising spatial regions associated with air traffic centers and altitude levels, in multiples of 3000' above mean sea level; forecast values are valid for six hour intervals; forecasts updated twice per day

Table 8 describes the generic component for an Information Domain Ontology. Ontologies have been used in computing for more than a decade, but only recently have they become commonplace. An ontology is a description of the semantic concepts of a domain, including important relations among those concepts. The simplest, most familiar ontologies come from biology, where Linnaean taxonomies describe plant and animal class relationships. Ontologies reduce the complexity of organizing facts. They also economize the recording of inferable facts. For example, we know all mammals have fur and nursing females lactate; therefore we can infer that our own female dog will lactate when she gives birth to puppies.

Our architectural component indicates that three attributes will be most important. We want to know the concepts addressed by an information source as well as the conditions it addresses. In the example, we can see how wind velocity and direction can be culled out of the data grid. With

additional ontology definitions, we can relate these conditions to others of interest, such as headwind component along a route segment and, ultimately, the fuel consumed and the reserve remaining. Standards for ontology representations are emerging, and we expect the information domain ontologies will become increasingly standardized. Prior to becoming standardized, however, we should expect that ontologies will evolve through use and experience among a community of practice. Each important operational problem requires information suppliers to meet the needs of planners and operators. This means that the ontologies will become increasingly adapted for use in condition monitoring and translation. The value of information, in short, derives from its ability to materially improve expected outcomes of operators' plans. Important and useful distinctions find their way into the concepts and conditions of the ontology, and these in turn determine the data that the suppliers report in their information sources.

Table 9. Operational Domain Ontology Generic Component Description.

FACET	VALUE	COMMENT
Name	**Operational Domain Ontology**	Same structure as Information Domain Ontology, but reflects operator concerns
Attribute	Concepts	Terms used and their semantic properties
Attribute	Conditions	Propositions and operators used to specify important situational characteristics
Attribute	Significant Deltas	Minimum changes in calculated conditions worthy of attention
Interface	Machine Interface	Planning Toolset can read attributes Condition Monitor can read attributes Domain Translator can read attributes
Interconnection	Planning Toolset reads attributes Condition Monitor reads attributes Domain Translator reads attributes	Planning tools use operational concepts to specify plans, assumptions, conditions Monitor determines which concepts, conditions and deltas pertain to its tasks Translator maps concepts and conditions from information domain to the operational domain
Quality	Ontology expressive and interpretable Deltas as small as necessary and as large as permissible	Language used should simplify writing/editing Machines can easily perform required inferences Appropriate deltas reduce workload
Example Use	Winds aloft concepts, part of aviation ontology Take-off fuel quantity Fuel consumption Adequate fuel reserve	Winds aloft described in terms of headwind and tailwind conditions along planned route of flight at planned time of flight; these decrement or increment airspeed to produce estimated groundspeed; groundspeed determines elapsed time for each route segment

Table 9 describes the generic component for the Operational Domain Ontology. This ontology is entirely similar to the ontology for the information domain, but it describes directly the concerns planners and operators have, rather than using the terms and codes of information suppliers. In the example, winds aloft are conceived in terms of their impact on groundspeed, flight time, fuel consumption, and the concern for adequate fuel reserve. Most operators today do the translation from information sources into operational domain concepts in their heads, routinely, often many times per

day. The VIRT architecture addresses the need to off-load such computation onto machines and to make it be "exception-driven" rather than intensive, repetitive, and usually immaterial. As the operational communities learn the value of making their concerns explicit, the planning tools will evolve to use the ontology concepts and conditions for human interface with the operators. In addition, dependencies will be converted into key conditions for monitoring. Lastly, the operational domain ontology will define the target range of the domain translator that can map source information into operational concerns.

Table 10. Domain Translator Generic Component Description.

FACET	VALUE	COMMENT
Name	**Domain Translator**	
Attribute	Conditions	Conditions the translator can infer in the target ontology
Attribute	Significant Deltas	Deltas the translator can infer in the target ontology
Interface	Machine Interface	Condition Monitor may employ translator
Interconnection	Condition Monitor employs	Condition Monitor evaluates operational conditions in part by inferring their values as translations of computed information domain values
Quality	Coverage Efficiency Correctness	Translations available for important conditions Machines can easily perform required inferences Translations and inferred values not erroneous
Example Use	Headwinds and tailwinds inferred from planned route, time, and winds aloft	The tailwind for each route segment is computed by finding the appropriate forecast wind aloft vector and computing the component parallel to the direction of flight; the headwind is the negative of the inferred tailwind

Table 10 describes the generic component for translating beliefs and values in one domain ontology into another. In many important applications, this is straightforward. Computations can be arranged either as goal-driven or change-driven. Goal-driven programs are asked to determine some values of a parameterized description, such as Headwind(route-segment-1, ?speed?). The interpreter of the ontology mapping then determines the actual value for the parameter ?speed? and returns an assertion with the parameter replaced by the actual speed along route-segment-1. Data-driven programs respond to changed observations and propagate inferences to the parties who have indicated a continuing need to be informed. Thus, when the twice-daily winds-aloft forecast is published, the headwind along each segment of each plan could be recomputed to determine if any significant deltas occurred. In that case, the new headwind value for the affected route segment would be conveyed to the appropriate parties.

Translation between formal languages has been a focus of computing research for decades. There are many simple to use language systems that can be used to build specialized translators. General-purpose, generic translators can also be built for a wide range of descriptive ontologies. The most general form of the translation problem is, in principle, not solvable, however. But that limitation isn't expected to have any practical impact on most applications of VIRT services, because these are likely to address practical domains where operators already do translation of this sort routinely, usually in their heads. Automating that work and doing it systematically for important conditions should produce significant value for planners and operators.

This completes the current description of the VIRT product-line architecture. We do not yet have much experience with actual implementations, and no off-the-shelf implementations exist for the

components. We are, thus, at the start of what should be a long-term, fruitful cycle of evolution, continuous improvement, and architectural refinement. The goal of framing an architecture at the outset is to encourage an approach that favors openness, reuse of assets, and a focus on quality components. These should make the benefits of VIRT available to more people, sooner, at lower cost.

Related Research

Many people have touched on aspects of model-based communication networks, adaptive replanning, information filtering, and selective information push. The principal related research is summarized briefly here.

Psychologists, sociologists, and students of decision-making and communication have demonstrated the importance of shared beliefs and shared context in interpersonal dialog and collaboration. We are all familiar with the phenomenon of communications becoming briefer among teammates, family members, and colleagues as familiarity and experience increases. In Shannon's original treatise on information theory, he characterized a single *bit* as the amount of information required to reduce 50% of the receiver's uncertainty. This means that the more communicating partners share beliefs, the less uncertainty they have, and the fewer bits they need to specify a preferred option.

In military and business operations, planners work to achieve similar communication efficiencies. They do this in multiple ways. First, they adopt specialized terms to characterize problem contexts, relevant potential actions and resources, and criteria by which possible plans should be judged. In addition to defining concepts embodying the key distinctions that clarify choices, they often adopt short-hand jargon. The specialized language and methods of communities of practice has recently been recognized as an important foundation for building effective systems.

Much of the shared context that simplifies communication between collaborators often consists of the perceived situation or its externalized representation. In the military, for example, we wish to provide all collaborators access to a *common operational picture (COP)* that portrays the battle space, actors, behaviors and intentions. In some human activities, the externalized representations have high "ecological validity." For example, in manufacturing, CAD/CAM technologies nearly guarantee that the "drawing" *is* the "part." In mountaineering and war, however, the "map is *not* the terrain." A COP is a constructed, compound, complex hypothesis. It never corresponds exactly to reality. Nevertheless, when teammates can see and share the same externalized representation, they can significantly reduce the volume of bits they exchange in order to designate an entity or characterize an option.

Several trends are pushing people beyond the point at which they can perform adequately in these contexts. First, the volume of potentially relevant information is increasing exponentially. Second, the required cycle times for adaptive response are shrinking. And third, the teammates increasingly are geographically distributed, often culturally dissimilar, and unfamiliar with one another. In these contexts, the informal methods that have enabled people to exchange a few bits in a face-to-face interaction with familiar colleagues won't suffice. Large portions of their collaborative work will have to be somewhat formalized so that computers can perform an increasing proportion of the information processing required.

Research on "human-machine symbiosis" traces its heritage to a famous paper by Licklider, who was the director of ARPA's Information Processing Techniques Office (IPTO). His vision continues alive today, and has already taken us through time sharing, personal computing with graphical user interfaces, the Internet and now the Web. Yet none of these developments have succeeded in enabling machines to off-load a great deal of the information filtering appropriate for planners and operators. This requires information consumers to clarify what they need to know and information suppliers to clarify what they provide. It then requires the machine to help translate from the source ontologies of suppliers into the operational ontologies of consumers. It's my belief that the foundations exist for all the capabilities presupposed in the architecture, but a concentrated effort needs to be undertaken to implement these and refine them to the point that communities of practice can regularly employ them.

Allowing operators and suppliers to "close the loop" is critical: ontologies, translations, and key conditions will all need to be refined through experience. Thus, all the tools need to be in the hands of the producers and consumers of information. Just as spreadsheet programs launched a revolution in business modeling and business use of interactive computing, I anticipate MCN and VIRT will launch a revolution in military and civilian use of ontologies and model-based information filtering for collaborative decision-making.

Principal Remaining Challenges

There are three principal challenges to making MCNs ubiquitous. First, we need practical ontologies for important domains. Second, we need leading operational communities to transform their processes around the "management-by-exception" style of VIRT. Third, we need open, evolutionary markets for information suppliers and consumers. While each of these has technological aspects, the more challenging aspects of each concern the managerial approach taken to business processes and the required transformations.

Before PCs and the rise of the Web, computer users expected to obtain systems from professional programmers. Systems were specified and acquired through sizable and difficult contracting arrangements. Some successful systems were procured this way. Many procurements failed. The major causes of failure were delay, cost overrun, and lack of usability or effectiveness. Simply stated, what computer users want is difficult to state, is often unknown to them, and usually changes over the course of months or years. Procurements are thus shooting at an ill-defined and moving target in many cases. Missing the target, then, should not be surprising.

The PC, with its personal applications such as spreadsheets, and the Web, with its ubiquitous authoring, hosting, and editing tools, have created a vibrant community of information suppliers and consumers. Many suppliers are professionals associated with businesses. Many suppliers are individuals. The trend is moving toward more suppliers, creating more sources, updating them more often: in short, more sources, more dynamism.

Organizations need to transform around the potential of dynamic information, agile response, distributed virtual enterprises, and self-synchronization, as in NCOW/IS. This means organizations must engage in continuous evolution, shifting their processes increasingly around better uses of information and computing. The "learning organization" is an excellent description of the new "business as usual" enterprise. MCNs enable the far-flung partners in an enterprise to collaborate succinctly, relying on externalized representations of common beliefs to eliminate the need for much redundant information exchange. VIRT services enable each planner and operator to off-load responsibility for continuous, repetitive review of important conditions to machines. All of this depends only on the development and continuous improvement of the ontologies, the shared models of how important things work in the domains of interest. Finally, an information market will enable new suppliers to proffer innovative information sources that VIRT condition monitors can automatically access. As part of such a market, whether commercial or controlled, sources need to be rated for quality and cost, so that consumers can exploit the most advantageous ones.

Organizations such as Navy FNMOC understand the importance of transforming from a traditional supplier of commodity information products to a key partner of operators who value important and timely information. FNMOC provides an excellent example of the leadership required to move into an important role in the network-centric future.

Near-term Exploitation Opportunities

I have identified several good opportunities for near-term exploitation of the VIRT services, including several with FNMOC as early collaborative partners in this effort. In essence, weather and oceanographic data are important to most soldiers, sailors and aviators. They examine copious amounts of data when generating plans, continually revalidating plans, and conducting operations. Many of these operations can achieve better outcomes with reduced risk if they can receive and exploit

improved forecasts or more accurate, timely updates. However, operators can't spend all their time looking at streams of dynamically updated weather data. For them to exploit the advantage of more and better information, they need vastly improved and automated filtering. VIRT is being applied to a number of applications within FNMOC. As one example, we aim to reduce the probability of detection of stealthy missions by assuring that the planners and operators receive valuable information at the right time.

As Ruokangas and Mengshoel demonstrated with their AWARE prototype, almost everyone in the aviation business can benefit from automated filtering and condition monitoring. Everyone who plans routes that are subject to unpleasant surprises would benefit from VIRT monitoring.

The DoD hopes to provide every level of command-control decision-making improved tools for situation awareness and real-time agile response. The COP is a foundation of this vision. The COP should be reconceptualized as a composite hypothesis of constituent models of the battle space and actors. Each component model, such as a blue-force flight or a neutral ship, can autonomously update its own expected state consistent with our knowledge of its plans and normal behavior. This obviates communication of unsurprising state changes. This allows the communication volume to be reduced to just those bits of *information*, corresponding to surprises or reductions in uncertainty. In this way, distributed collaborators can achieve a higher level of shared understanding with reduced volumes of communication. This, in turn, means they can spend more time on high-value activities, rather than being kept busy processing low-value data.

Organizations that exist to supply important information to planners and operators have a great opportunity to begin moving into the new paradigm that the VIRT architecture describes. Their products will be more valued if they are characterized by ontologies and if these are related to and translatable into operational domain ontologies. In the case of weather, as an example, the most successful organizations should be measuring their progress in terms of the import, ease, efficiency, and timeliness planners and operators attribute to their products. These are the kinds of ratings that will become advertisements and evaluation criteria as the open information market develops. Being first and best at the new game can establish significant, potentially permanent, competitive advantages and leadership.

In short, the best opportunities will arise with the organizations most eager to accelerate the transformation into net-centric, information-superior enterprises. As with other generic technologies, the real question isn't whether VIRT is applicable, but "Who's ready now?"

Summary and Conclusion

This paper has described a vision of a transformed way of operating, especially for organizations that routinely plan and execute plans. The need for this transformation arises from both new problems and new opportunities. The new problems concern new kinds of competitive challenges and new pressures to behave with greater speed, agility, and precision. As the types and volume of potentially relevant information increase without bounds, the pressures on humans to produce excellent decisions and outcomes become unrealistic. Humans need to exploit computing power to reduce their tasks to a manageable level. For our organizations to get the best results, the human resources need to spend their limited time on the most important things. MCN and VIRT provide frameworks for doing that. This new architecture exploits several significant opportunities that have been developed over the last decade: (1) networked communication; (2) ontologies and inference; (3) information filtering by machines; and (4) incredibly cheap computers.

The architecture proposed here must be implemented in specialized applications with particular supplier and operator communities to prove its worth and thus become "obvious" to a larger population. As with many new "obvious" technologies, the early successes require leadership and pioneering experiments. Some of this is now underway, but much more needs to be done. The point of this paper is to provide a simple trail map for pioneers to follow.

References

1. Coram, R., *Boyd: The fighter pilot who changed the art of war*. 2002, Boston: Little, Brown.
2. Erman, L.D., et al., *The Hearsay-II Speech-understanding system: Integrating knowledge to resolve uncertainty*. Computing Surveys, 1980. **12**(2).
3. Group, O.R.-T.S., *Data Distribution Service for Real-Time Systems Specification*, in *OMG Adopted Specification*. 2003, Object Mgt. Group.
4. Lesser, V.R. and D.D. Corkill, *Functionally-accurate, cooperative distributed systems.* IEEE Transactions on Systems, Man, and Cybernetics, 1981. **SMC-11**: p. 81-96.
5. Wesson, R., et al., *Network structures for distributed situation assessment*, in *Readings in Distributed Artificial Intelligence*, A. Bond and L. Gasser, Editors. 1981, Morgan Kaufmann. p. 71-89.
6. Alberts, D.S., J.J. Garstka, and F.P. Stein, *Network Centric Warfare: Developing and Leveraging Information Superiority*. 2nd ed. 2002, Washington, D.C.: Office of the Secretary of Defense (ASD/C3I/CCRP).
7. Hayes-Roth, F., *Using proofs and refutations to learn from experience*, in *Machine Learning*, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, Editors. 1983, Tioga Publishing: Palo Alto, CA. p. 221-240.
8. Ruokangas, C.C. and O.L. Mengshoel. *Information filtering using bayesian networks: effective user interfaces for aviation weather data*. in *Proceedings of the 8th international conference on Intelligent user interfaces*. 2003: ACM.
9. Bosch, J., *Design and use of software architectures*. 2000: Addison-Wesley.
10. Clements, P., R. Kazman, and M. Klein, *Evaluating software architectures: methods and case studies*. 2002: Addison-Wesley.